The Network Layer!

- Application: the application (e.g., the Web, Email)
- Transport: end-to-end connections, reliability
- Network: routing
- Link (data-link): framing, error detection
- Physical: 1’s and 0’s/bits across a medium (copper, the air, fiber)
IP Datagrams

- IP Datagrams are like a letter
  - Totally self-contained
  - Include all necessary addressing information
  - No advanced setup of connections or circuits

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>8</th>
<th>12</th>
<th>16</th>
<th>19</th>
<th>24</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>HLen</td>
<td>DSCP/ECN</td>
<td>Datagram Length</td>
<td>Identifier</td>
<td>Flags</td>
<td>Offset</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Header Checksum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source IP Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination IP Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options (if any, usually not)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data (variable len: typically TCP/UDP segment)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How does an end host get an IP address?

• Static IP: hard-coded
  – Windows: control-panel->network->configuration->tcp/ip->properties
  – UNIX: /etc/rc.config

• DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  – “plug-and-play”
DHCP: Dynamic Host Configuration Protocol

Goal: allow host to dynamically obtain its IP address from network server when it joins network
  – can renew its lease on address in use
  – allows reuse of addresses
  – support for mobile users who want to join network

DHCP overview:
  – host broadcasts “DHCP discover” msg [optional]
  – DHCP server responds with “DHCP offer” msg [optional]
  – host requests IP address: “DHCP request” msg
  – DHCP server sends address: “DHCP ack” msg
DHCP client-server scenario

DHCP server: 223.1.2.5

DHCP discover
src: 0.0.0.0, 68
dest: 255.255.255.255, 67
yiaddr: 0.0.0.0
transaction ID: 654

DHCP offer
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 654
lifetime: 3600 secs

DHCP request
src: 0.0.0.0, 68
dest: 255.255.255.255, 67
yiaddr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs

DHCP ACK
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs
DHCP: More than IP Addresses

DHCP can return more than just allocated IP address on subnet:

- address of first-hop router for client (default GW)
- name and IP address of DNS server(s)
- subnet mask
IP Fragmentation, Reassembly

- Higher layer’s data unit is too large for the lower layer
- Fragmentation: taking a large data unit and breaking it into smaller chunks
- Assembly: combining chunks into the original data unit.

Examples:
- Transport: TCP takes stream of bytes and breaks into TCP segments
- Network: IP takes packets too big for a link and breaks them up into IP fragments
- Link: 6lowpan takes IPv6 packets and breaks them into link fragments if needed.
Different link layers have different MTUs (max transfer size) - largest possible link-level frame

large IP datagram divided ("fragmented") into several datagrams

\[ \text{fragmentation:} \quad \text{in: one large datagram} \]
\[ \text{out: 3 smaller datagrams} \]

\[ \text{reassembly} \]

\[ \text{fragmentation:} \quad \text{each smaller fragment is routed independently} \]
### IP Datagram Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4</td>
</tr>
<tr>
<td>HLen</td>
<td>8</td>
</tr>
<tr>
<td>Type of Service</td>
<td>12</td>
</tr>
<tr>
<td>Datagram Length</td>
<td>16</td>
</tr>
<tr>
<td>Identifier</td>
<td>16</td>
</tr>
<tr>
<td>DF</td>
<td>19</td>
</tr>
<tr>
<td>MF</td>
<td>24</td>
</tr>
<tr>
<td>Offset</td>
<td>31</td>
</tr>
<tr>
<td>TTL</td>
<td></td>
</tr>
<tr>
<td>Protocol</td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td></td>
</tr>
<tr>
<td>Source IP Address</td>
<td></td>
</tr>
<tr>
<td>Destination IP Address</td>
<td></td>
</tr>
<tr>
<td>Options (if any, usually not)</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>

- **Version**: Used to identify the version of the IP protocol being used.
- **HLen**: Header length, indicating the length of the header in 32-bit words.
- **Type of Service**: Reflects the service level of the packet.
- **Datagram Length**: The total length of the packet in bytes.
- **Identifier**: Used to identify which larger chunk a fragment belongs to.
- **DF (Don’t Fragment)**: Flags if last fragment.
- **MF (More Fragments)**: Offset field to piece fragments together in order.
IP Fragmentation, Reassembly

- Different link layers have different MTUs (max transfer size) - largest possible link-level frame
- Large IP datagram divided (“fragmented”) into several datagrams
  - Reassembled only at final destination
  - IP header bits used to identify, order related fragments
IP Fragmentation, Reassembly

- IP addresses plus ident field identify fragments of a packet
- MF bit is 1 in all but last fragment
- Offset field says location of fragment (in 8 byte chunks)
  - All fragments except last one must be multiple of 8 bytes long

<table>
<thead>
<tr>
<th>Before fragmentation</th>
<th>After fragmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>start of header</td>
<td>start of header</td>
</tr>
<tr>
<td>ident=x</td>
<td>ident=x</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>offset=0</td>
<td>offset=0</td>
</tr>
<tr>
<td>rest of header</td>
<td>rest of header</td>
</tr>
<tr>
<td>1400 bytes</td>
<td>512 bytes</td>
</tr>
<tr>
<td>start of header</td>
<td>start of header</td>
</tr>
<tr>
<td>ident=x</td>
<td>ident=x</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>offset=64</td>
<td>offset=64</td>
</tr>
<tr>
<td>rest of header</td>
<td>rest of header</td>
</tr>
<tr>
<td>512 bytes</td>
<td>512 bytes</td>
</tr>
<tr>
<td>start of header</td>
<td>start of header</td>
</tr>
<tr>
<td>ident=x</td>
<td>ident=x</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>offset=128</td>
<td>offset=128</td>
</tr>
<tr>
<td>rest of header</td>
<td>rest of header</td>
</tr>
<tr>
<td>376 bytes</td>
<td></td>
</tr>
</tbody>
</table>
IP Path MTU Discovery

- Avoid fragmentation: Host tests link with a large packet
- Implemented with ICMP: set DF – do not fragment. Triggers error response from a router
Recall: IPv4 Addresses

• 32-bit number, must be globally unique

• $2^{32} \Rightarrow 4,294,967,296$ possible addresses

• How many do you have?
ARIN Finally Runs Out of IPv4 Addresses

IPv4 Address Cupboards are Bare in North America.

It is often said, "the Internet is running out of phone numbers," as a way to express that the Internet is running out of IPv4 addresses, to those who are unfamiliar with Internet technologies. IPv4 addresses, like phone numbers are assigned hierarchically, and thus, have inherent inefficiency. The world’s Internet population has been growing and the number of internet-connected devices continues to rise, with no end in sight. In the next week, the American Registry for Internet Numbers (ARIN) will have exhausted their supply of IPv4 addresses. The metaphorical IPv4 cupboards are bare. This long-predicted Internet historical event marks opening a new chapter of the Internet’s evolution. However, it is somehow anti-climactic now that this date has arrived. The Internet will continue to operate, but all organizations must now accelerate their efforts to deploy IPv6.

ARIN IPv4 Address Exhaustion

The Internet Assigned Numbers Authority (IANA) delegates authority for Internet resources to the five RIRs that cover the world. The American Registry for Internet Numbers (ARIN) is the Regional Internet Registry (RIR) for the United States, Canada, the Caribbean, and North Atlantic islands. ARIN has been managing the assignment of IPv4 and IPv6 addresses and Autonomous System (AS) numbers for several decades. Each RIR has been managing their limited IPv4 address stores and going through their various phases of exhaustion policies. ARIN has been in Phase 4 of their IPv4 depletion plan for more than a year now. ARIN will soon announce that they have completely extinguished their supply of IPv4 addresses.

Insider

Network jobs are hot; salaries expected to rise in 2016

Related Topics

Internet
Cisco Subnet
IPv6

Comments

An insider’s guide to the private IPv4 market
Techniques for Prolonging the Lifespan of IPv4
ARIN’s registry and transfer policies can help bridge the gap from IPv4 to IPv6
If I buy a Chromebook and can’t get to grips with OS can I convert to windows?
Seriously, we're done now. We're done

Exhausted with never-ending internet exhaustion

By Kieren McCarthy in San Francisco 15 Feb 2017 at 23:07 214 ▼ SHARE ▼

You may have heard this before, but we are really, really running out of public IPv4 addresses.

This week, the regional internet registry responsible for Latin America and the Caribbean, LACNIC, announced it has moved to "phase 3" of its plan to dispense with the remaining network addresses, meaning that only companies that have not received any IPv4 space are eligible. There is no phase 4.

That means LACNIC is down to its last 4,698,112 public IPv4 addresses (although that may increase as it recovers a little bit of space over time).
OK, this time it's for real: The last available IPv4 address block has gone

Now for the last time, will you all please shift to IPv6?!

By Kieren McCarthy in San Francisco 18 Apr 2018 at 22:10

You may have heard this one before, but we have now really run out of public IPv4 address blocks.

The Internet Assigned Numbers Authority – the global overseers of network addresses – said it had run out of new addresses to dish out to regional internet registries (RIRs) in 2011. One of those RIRs, the Asia-Pacific Network Information Centre, said it was out of available IPv4 addresses later that year.

Then Europe's RIR, Réseaux IP Européens aka RIPE, ran dry in September 2012, followed by the Latin America and Caribbean Network Information Centre (LACNIC) in June 2014. Next, the American Registry for Internet Numbers hit an IPv4 drought in September 2015.
Private Addresses

• Defined in RFC 1918:
  – 10.0.0.0/8   (16,777,216 hosts)
  – 172.16.0.0/12 (1,048,576 hosts)
  – 192.168.0.0/16 (65536 hosts)

• These addresses shouldn’t be routed.
  – Anyone can use them.
  – Often adopted for use with NAT.
NAT: Network Address Translation

all datagrams leaving local network have same single source NAT IP address: 138.76.29.7, different source port numbers

datagrams with source or destination in this network have 10.0.0/24 address for source, destination
Implementing NAT

• Two hosts communicate with same destination
  – Destination needs to differentiate the two
• Map outgoing packets
  – Change source address and source port
• Maintain a translation table
  – Map of (src addr, port #) to (NAT addr, new port #)
• Map incoming packets
  – Map the destination address/port to the local host
NAT: network address translation

**1:** host 10.0.0.1 sends datagram to external server, 80

**2:** NAT router changes datagram source addr from local address, 3000 to NAT address, 7000, updates table

**3:** reply arrives dest. address: NAT address, 7000

**4:** NAT router changes datagram dest addr from NAT Address, 7000 to Local Address, 3000
Neither the sender nor receiver need to know that NAT is happening...

4: NAT router changes datagram dest addr from NAT Address, 7000 to Local Address, 3000
NAT Advantages

• Organizations need fewer IP addresses from their ISP.
  – With a 16-bit port field, we can put 65535 connections behind one external IP address!

• Organizations can change internal network IPs without having to change outside world IPs.
Principled Objections Against NAT

• Routers are not supposed to look at port #s
  – Network layer should care only about IP header
  – ... and not be looking at the port numbers at all

• NAT violates the end-to-end argument
  – Network nodes should not modify the packets

• IPv6 is a cleaner solution
  – Better to migrate than to limp along with a hack

That’s what happens when network puts power in hands of end users!
IPv6

• **Initial motivation:** 32-bit address space soon to be completely allocated, any day now™.

• **Additional motivation:**
  – header format helps speed processing/forwarding
  – header changes to facilitate QoS

**IPv6 datagram format:**
  – fixed-length 40 byte header
  – no fragmentation allowed
IPv6 Header

- Double the size of IPv4 (320 bits vs. 160 bits)

---

**IPv6 Header Diagram**

```
0  4  8  12  16  19  24  31
```

<table>
<thead>
<tr>
<th>Version</th>
<th>Type of service</th>
<th>Flow Label</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Datagram Length</th>
<th>Next Header</th>
<th>Hop Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source IP Address (128 bits)

Destination IP Address (128 bits)

Data (variable len: typically TCP/UDP segment)

Groups packets into flows, used for QoS

Same as TTL in IPv4
Other changes from IPv4

- **checksum**: removed entirely to reduce processing time at each hop
- **options**: allowed, but outside of header, indicated by “Next Header” field
- **ICMPv6**: new version of ICMP
  - additional message types, e.g. “Packet Too Big”
  - multicast group management functions
IPv6 (vs. IPv4)

• Simpler, faster, better

• How much traffic on the Internet is IPv6?

• Why?!

IPv6 celebrates its 20th birthday by reaching 10 percent deployment

All I want for my birthday is a new IP header.

ILJITSCH VAN BEIJNUM - 1/3/2016, 12:00 PM

Twenty years ago this month, RFC 1883 was published: Internet Protocol, Version 6 (IPv6) Specification. So what’s an Internet Protocol, and what’s wrong with the previous five versions? And if version 6 is so great, why has it only been adopted by half a percent of the Internet’s users each year over the past two decades?

10 percent!

First the good news. According to Google’s statistics, on December 26, the world reached 9.98 percent IPv6 deployment, up from just under 6 percent a year earlier. Google measures IPv6 deployment by having a small fraction of their users execute a Javascript program that tests whether the computer in question can load URLs over IPv6. During weekends, a tenth of Google’s users are able to do this, but during weekdays it’s less than 8 percent. Apparently more people have IPv6 available at home than at work.
Transitioning to IPv6

• Option 1: “Flag day”
  – How do we get everyone on the Internet to agree?
  – Whose authority to decide when?
  – Can you imagine how much would break?

• Option 2: Slow transition
  – Some hosts/routers speak both versions
  – Must have some way to deal with those who don’t
  – Lack of incentive to switch
Tunneling

- IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers
Tunneling

**logical view:**

```
A: IPv6 -> B: IPv6 -> C: IPv6 (IPv4 tunnel)
```

**physical view:**

```
A: IPv6 -> B: IPv6 -> C: IPv4 -> D: IPv4 -> E: IPv6 -> F: IPv6
```

**Flow:**

- **src:** A, **dest:** F
- **src:** B, **dest:** E

**Data packets:**

- **A-to-B:** IPv6
- **B-to-C:** IPv6 inside IPv4
- **B-to-C:** IPv6 inside IPv4
- **E-to-F:** IPv6
ICMP: Internet Control Message Protocol

• Service Model
  – Reporting message: self-contained message reporting error
  – Unreliable: Simple datagram service – no retries.
ICMP: Internet Control Message Protocol

• Used to communicate network information
  – “Control messages”, i.e., not data themselves
  – Error reporting
    • Unreachable host
    • Unreachable network
    • Unreachable port
    • TTL expired
  – Test connectivity
    • Echo request/response (ping)
ICMP: Internet Control Message Protocol

- Header:
  - 1-byte type
  - 1-byte code
  - 2-byte checksum
  - 4 bytes vary by type

- Sits above IP
  - Type 1 in IP header
  - Usually considered part of IP
### ICMP: Internet Control Message Protocol

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>echo reply (ping)</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>dest. network unreachable</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>dest host unreachable</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>dest protocol unreachable</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>dest port unreachable</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>dest network unknown</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>dest host unknown</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0</td>
<td>source quench (congestion control - not used)</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>echo request (ping)</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>route advertisement</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>router discovery</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>TTL expired</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>bad IP header</td>
</tr>
</tbody>
</table>

![Diagram showing the structure of an ICMP message]
Ping
Traceroute and ICMP

- Source sends sets of UDP segments (usually 3) to dest
  - first set has TTL = 1
  - second set has TTL = 2, etc.
  - unlikely port number
- When $n$th set of datagrams arrives to $n$th router:
  - router discards datagrams
  - and sends source ICMP messages (type 11, code 0)
  - ICMP messages includes name of router & IP address
- When ICMP messages arrives, source records RTTs
Traceroute and ICMP

stopping criteria:

• UDP segment eventually arrives at destination host
• destination returns ICMP “port unreachable” message (type 3, code 3)
• source stops
Traceroute Demo

Te = Time exceeded
Pu = Port unreachable

TTL=1, Dest = B, port = invalid
Te (R1)

TTL=2, Dest = B

TTL=3, Dest = B
Te (R2)

TTL=4, Dest = B
Te (R3)

Pu (B)